O'REILLY®

oscon

open source convention

oscon.com

Architectural Anti Patterns

Notes on Data Distribution and Handling Failures

Gleicon Moraes

$whoami

work: http://locaweb.com

site: http://7co.cc

blog: http://zenmachine.wordpress.com

code: http://github.com/gleicon

RestMQ: http://restmq.com

twitter: @gleicon

Motivation

- User group meetings with friends to NoSQL:br, QConSP and now OSCon Data !
- Consulting cases
- Large volumes of data / High concurrent access
- Distribution and Disaster recovery
- Architecture
- Operations
- Live migrations

Required Listening

Tracklist

- Elephant Talk
- Frame by Frame
- Matte Kudasai (please wait)
- Indiscipline
- Thela Hun Ginjeet (heat in the jungle)
- The Sheltering Sky
- Discipline
- Matte Kudasai (Bonus Track – alternate mix)

Agenda

- Elephant Talk - Why Anti Patterns ?
- Frame by Frame - DML/DDL Anti Patterns
- Matte Kudasai (please wait) - Architectural Anti Patterns
- Indiscipline - Abusing the RDBMS
- Thela Hun Ginjeet (Heat in the jungle) - A word about Ops
- The Sheltering Sky - Architecture and Migration
- Discipline - The discussion over data organization + Questions

Elephant Talk
(Why Anti Patterns ?)

Why Anti Patterns

Architectural Anti Patterns for Data Handling or 'Please don't do it with [MySQL|PGSQL|Oracle|SQLServer]' or 'Things that you shouldn't use a RDBMS for'.

Still, why Anti Patterns ?

- Mostly result of innocent assumptions
- You catch them at the server crashing or customer complaints
- NoSQL databases are not safe from it. Mistaken assumptions are like water, finds a way around everything.

Quick Question

“What is the main MySQL tuning technique used in panic situations ?”

Quick Question

“What is the main MySQL tuning technique used in panic situations ?”

A: To install PostgreSQL

RDBMS Anti Patterns

News Flash: Not everything fits on a RDBMS.

The Great Architectural Staple (tm), used as:

- Message Queues
- Job Schedulers
- Distributed Counters
- Distributed Lock
- Document Storage
- Filesystem
- Cache
- Logger
- Workflow Orchestrator

Frame by Frame
(DML/DDL Anti Patterns)

The Eternal Tree

Description

Queries around a single table, self referencing an id or key.

Example

Most threaded discussion examples uses something like a table which contains all threads and answers, relating to each other by an id.

Discussion

At any given point the application code will contain its own tree-traversal algorithm to manage this.

Also most cases have a limit in depth for performance reasons. Any kind of sharding requires that all comments be at the same instance.

The Eternal Tree

Table

ID	Parent-ID	Author	Text
1	0	gleicon	Hello World
2	1	elvis	Shout!

Same data stored as a document

```
{ thread_id:1, title: 'the meeting', author: 'gleicon', replies:[
      {
        'author': elvis, text:'shout', replies:[{...}]
      }
   ]
}
```

The same sharding and indexing issues still apply.

Dynamic Table Creation

Discussion

To avoid huge tables, one must come with a "dynamic schema".

Example

A Document storage/management company, which is adding new facilities over the country. For each storage facility, a new table is created, structured like this:

Item_id	Row	Column	Description
1	10	20	Cat food
2	12	32	Trout

Discussion

To query the total capacity of a set of facilities over a given state, one query to an index table would have to be issued, and then one query for each facility on its own table.

The Alignment

Description

Extra columns are created but not used immediately. Usually they are named as a1, a2, a3, a4 and called padding.

Example

Data with variable attributes or highly populated tables which broke on migrations when adding an extra column in a 150M lines database and it took 2 days to run an ALTER TABLE

Discussion

Having no used columns might feel hackish as when one used padding on ANSI C structs, but probably the best alternative to variable attributes is a document based database.

Specially if the attributes are local to each document

Hierarchical Sharding

Description

Using databases to shard data inside the same server/instance.

Example

Email accounts management. Distinct databases inside a RDBMS ranging from A to Z, each database has tables for users starting with the proper letter. Each table has user data.

```
> show databases;
a b c d e f g h i j k l m n o p q r s t u w x z
> use a
> show tables;
...alberto alice alma ... (and a lot more)
```

Hierarchical Sharding

Discussion

There is no way to query anything in common for all users without application side processing. JOINs are not possible.

RDBMS have all tools to deal with this particular case of 'different clients and their data'

The letters A and M (at least for Brazilian Portuguese) will alone have more names than the rest.

Embedded Lists

Description

As data complexity grows, one thinks that it's proper for the application to handle different data structures embedded in a single cell or row.

Example

Flags for feature administration

```
> select group_field from that_email_sharded_database.user
"a@email1.net, b@email1.net,c@email2.net"

> select flags from email_feature_admin where id = 2
1|0|1|1|0|
```

Embedded Lists

Discussion

The popular 'Let's use commas to separe it'. You may find distinct separators as |, -, [] and so on. Embedding CSV fields would fall on this category

Apart from space optimization, a very common argument is to save space.

A proper alternative is a clone of proper SETS and LISTS on K/V stores as Redis, acting as a property storage. Or even a row for each flag.

Matte Kudasai

(Architectural Anti Patterns)

Heroes are hard to find

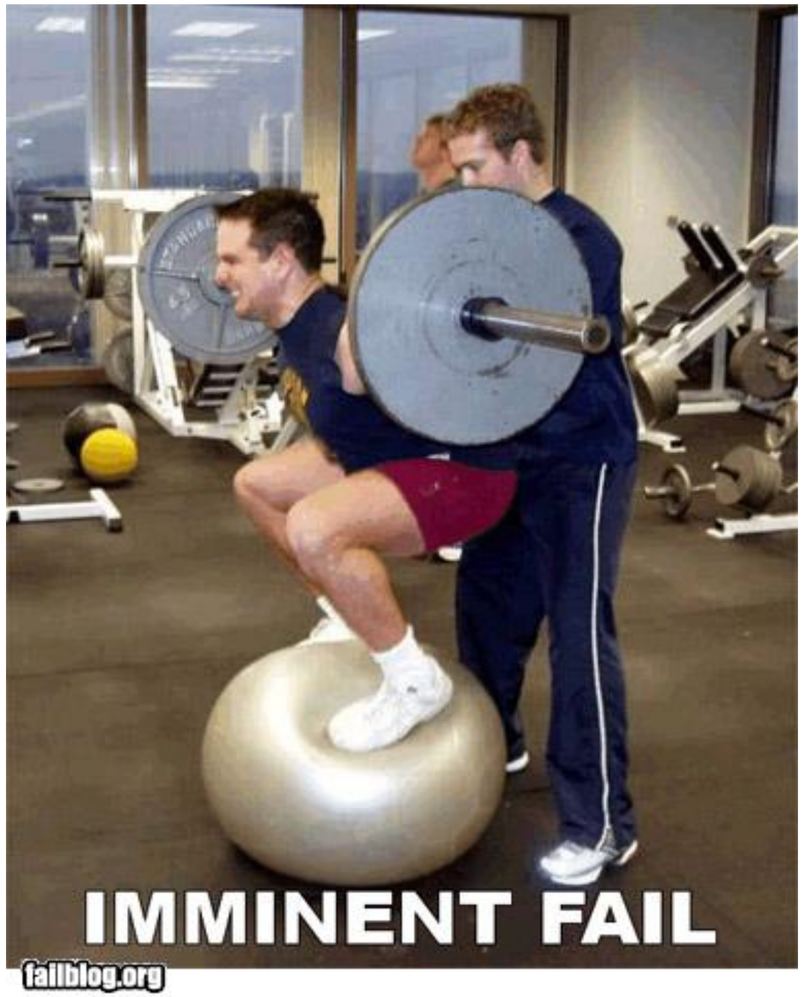


Table as Cache

Description

A workaround for complex queries which stores resultsets in a different table using the query - or part of it as a key.

Discussion

Complex and recurrent queries that demand the result to be stored in a separated table, so they result can be retrieved quickly.

Queries with many JOINs or complex data dictionaries, in late stage development or legacy patching.

Also, it de-normalizes the data and demands cache maintenance as purging and invalidation.

In-memory cache vs DB backed cache access time

Table as Queue

Description

The use of a table, or more than one table, to hold messages to be processed, instead of a Message Broker.

Discussion

Table/views with rows ordered by time of creation

Demands application polling and cleaning

Race condition when more than one client is polling

Wrapped on stored procedures/triggers to signal new items

BONUS: global procedure to generate job/message IDs

Table as Log File

Description

Logs written straight to one or more tables fully or in a round-robin fashion.

Discussion

Concurrent access in the same schema than application data

From time to time it needs to be purged/truncated

Ordered/Indexed by date, to avoid grep'ing log files

Usually application/transaction log

Scheme only fits in an A0 sheet

Description

High specialized and normalized data dictionaries

Example

The product description is local to a 'child' table, but the main expiration ref is represented by the main product table.

Main_Product_Table

ID	Dept	Due_date	Expires_in

Product_A

ID	Desc

Product_B

ID	Desc

Scheme only fits in an A0 sheet

Discussion

Extreme specialization creates tables which converges to key/value model.

To model the data scheme after modeling the applications objects

The normal form get priority over common sense to avoid de-normalization

Extreme (as in extreme sports) JOINs required and views to hide the complexity.

Your ORM

Description

Expectations over the use of an ORM over database performance and best practices.

Discussion

Your ORM issue full queries for each dataset iteration

Performance is not what was expected as the data volume grows

ORMs are trying to bridge two things with distinct impedance (domain(set) of each column -> Objects atributes)

Indiscipline
(Abusing the RDBMS)

Stoned Procedures

Description

On the use of stored procedures to hold application logic and event signaling outside of the database context. Triggers and Stored procedures combined to orchestrate applications

Discussion

Stored Procedures and Triggers has the magic property of vanishing of our memories and being impossible to keep versioned. Also, they are hard to test properly.

Event handling on different architecture piece

Bundled Map/Reduce as stoned procedures for non-RDBMS ?

Distributed Global Locking

Description

On trying to emulate JAVA's synchronize in a distributed manner, initially as a ref counter

Example

```
> select COALESCE(GET_LOCK('my_lock',0 ),0 )
```

Discussion

RDBMS are not the proper place to locking (or ref counters, which may act as soft locks)

Embedded in a magic classes called DistributedSynchronize, ClusterSemaphore, DistributedCluster, SynchronizedSemaphore

Throttle Control

Description

On delegating to the database the control and access tracking to a given resource

Example

A sequence of statements is issued, varying from a update...select to a transaction block using a stored procedure:

> select count(access) from throttle_ctl where ip_addr like ...
> update .... or begin ... commit

Throttle Control

Discussion

For each request would have to check on a transaction block.

Usually mixed with a table-based ACL.

Using memcached (or any other k/v store) to add and set expire time:

```
if (add 'IPADDR:YYYY:MM:DD:HH', 1) < your_limit: do_stuff()
```

Theela Hun Ginjeet

(A word about Ops)

A word about operations

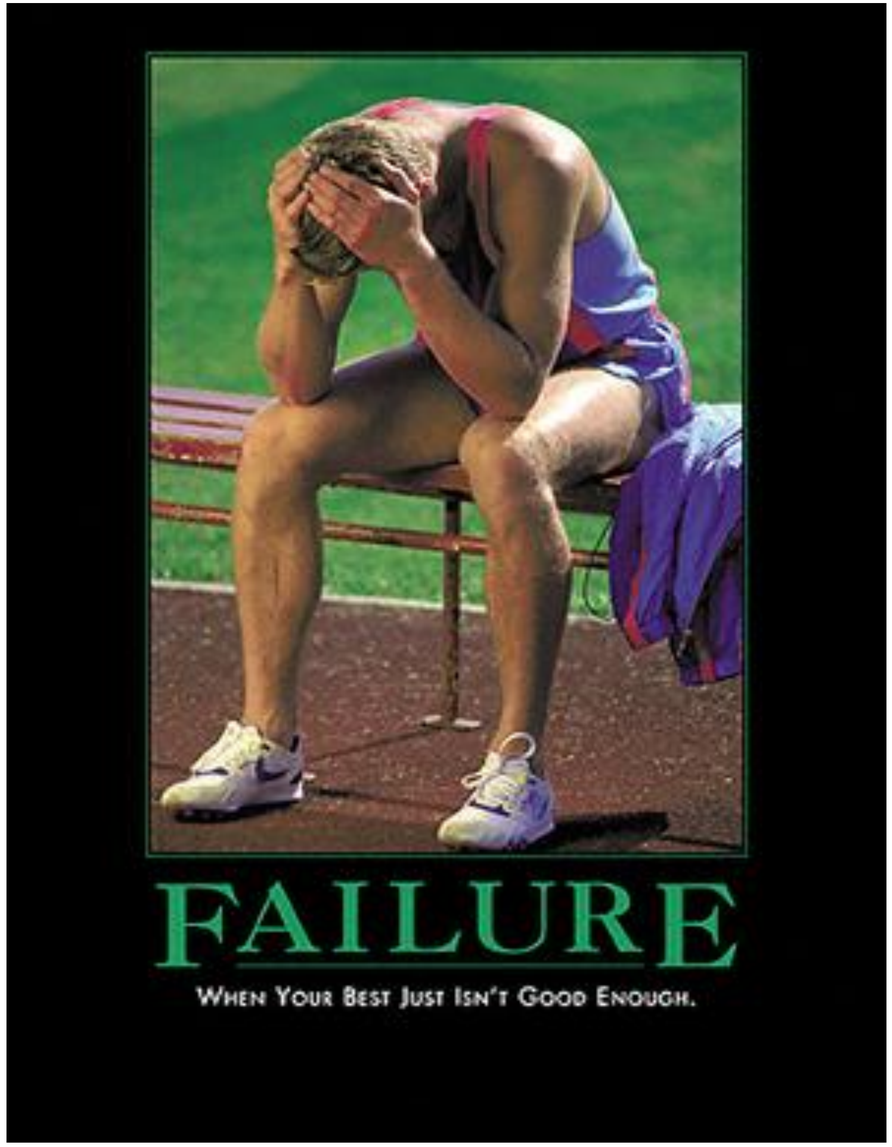
The obligatory quote ritual

"Without time constraints, any migration is easy, regardless data size"
(Out of production environment, events flow in a different time)

Email Data stored on Databases

- Email traffic accounting ~4500 msgs/sec in, ~2000 msgs out
- 10 Gb SPAM/NOT SPAM tokenized data per day
- 300 Gb logfile data/day
- 80 Billion DNS queries per day / 3.1 GB/s traffic
- ~1k req/sec tcptable queries.
- 0.8 Pb of data migrated over mixed quality network.
- Application logs on MySQL / High concurrent data on PostgreSQL

Fail seven times, stand up eight



The Sheltering Sky
(Architecture and Migration)

Cache

Think if the data you use aren't already de-normalized somewhere (cached)

Are you dependent on cache ? Does your application fails when there is no warm cache ? Does it just slows down ? MTTR ?

Migration

How to put and get data back from the database ?
SQL queries, CSV, binary file, HTTP, Protocol Buffers

How to change the schema ? Is there a schema ?

Rough calculation to stream 1 Terabyte over a 1 Gigabit/sec link, assuming you have a dedicated network segment for that and constant throughput of 70% (~ 716.8 Mbps/sec): 3h15m

Not considering:

- protocol overhead
- concurrent traffic (live migrations)
- indexing

Architecture

Most of the anti-patterns signals that there are architectural issues instead of only database issues.

Call it NoSQL, non relational, or any other name, but assemble a toolbox to deal with different data types.

Discipline
(Data Organization)

The obligatory quote ritual

"Without operational requirements, it's a personal choice about data organization"

Data Organization - Storing

MongoDB

```
{
 'id': 1,
 'title' : 'Diving into Python',
 'author': 'Mark Pilgrim',
 'tags': ['python','programming', 'computing']
}

{
 'id':2,
 'title' : 'Programing Erlang',
 'author': 'Joe Armstrong',
 'tags': ['erlang','programming', 'computing',
'distributedcomputing', 'FP']
}

{
 'id':3,
 'title' : 'Programing in Haskell',
 'author': 'Graham Hutton',
 'tags': ['haskell','programming', 'computing', 'FP']
}
```

Redis

```
SET book:1 {'title' : 'Diving into Python',
 'author': 'Mark Pilgrim'}
SET book:2 { 'title' : 'Programing Erlang',
 'author': 'Joe Armstrong'}
SET book:3 { 'title' : 'Programing in Haskell',
 'author': 'Graham Hutton'}


SADD tag:python 1
SADD tag:erlang 2
SADD tag:haskell 3
SADD tag:programming 1 2 3
SADD tag computing 1 2 3
SADD tag:distributedcomputing 2
SADD tag:FP 2 3
```

Data Organization - Querying

MongoDB

Search tags for erlang or haskell:

```
db.books.find({"tags":
    { $in: ['erlang', 'haskell']
  }
})
```

Search tags for erlang AND haskell (no results)

```
db.books.find({"tags":
    { $all: ['erlang', 'haskell']
  }
})
```

This search yields 3 results

```
db.books.find({"tags":
    { $all: ['programming', 'computing']
  }
})
```

Redis

Search tags for erlang and haskell:

```
SINTER 'tag:erlang' 'tag:haskell'
0 results
```

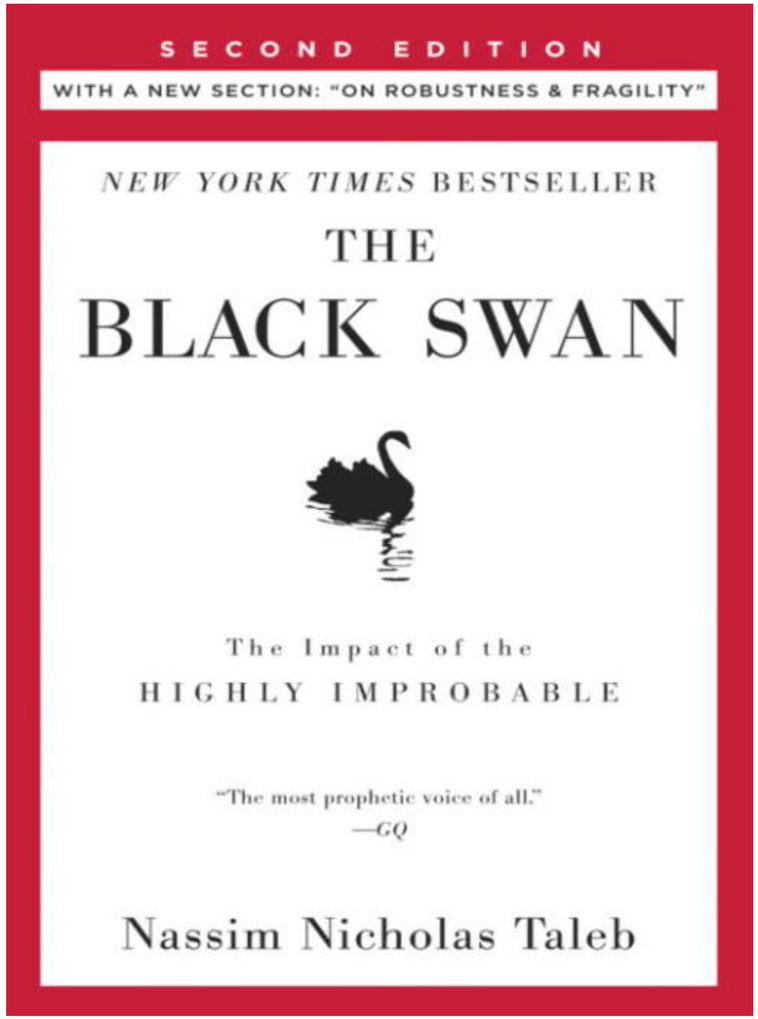
Search for programming and computing tags:

```
SINTER 'tag:programming' 'tag:computing'
3 results: 1, 2, 3
```

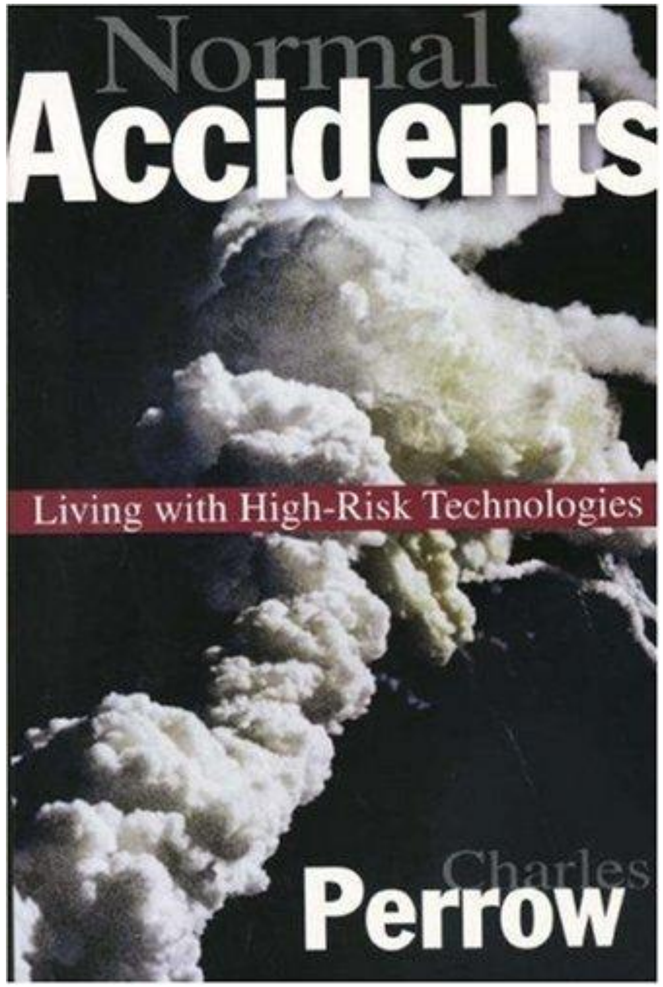
```
SUNION 'tag:erlang' 'tag:haskell'
2 results: 2 and 3
```

```
SDIFF 'tag:programming' 'tag:haskell'
2 results: 1 and 2 (haskell is excluded)
```

Obligatory book ref



Obligatory book ref



Questions ?

Thanks !